SGRT: A Scalable Mobile GPU Architecture based on Ray Tracing

Won-Jong Lee†, Shi-Hwa Lee†, Jae-Ho Nah*, Jin-Woo Kim*, Youngsam Shin†, Jaedon Lee†, Seok-Yoon Jung†

SAIT, SAMSUNG Electronics†, Yonsei Univ.*, Korea
Outline

• Introduction

• SGRT Core Architecture
  – T&I Engine: H/W Accelerator
  – SRP: Programmable DSP
  – SMK: Parallelization Framework

• Experimental Results

• Conclusion
Introduction
Graphics Trends

- Graphics is being important as increasing smart devices
- Evolving toward more realistic graphics
- Mobile graphics template earlier PC graphics (5~6 years)
Mobile SoC

Apple A5X Die Photo
Image Courtesy: Chipworks
Current Mobile GPU for Ray Tracing

• Inadequate Performance
  – Flagship mobile GPU: $\approx 256$GFLOPS (ARM Mali T658)
  – Real-time ray tracing @HD: $>300$Mray/sec (1~2TFLOPS)

• Unsuitable Execution Model
  – “Multithreaded SIMD” is not fit for processing incoherent rays

• Weak Branch Supports
  – Performance drops when recursion, function calls, control flow…
Need a New Architecture?

- Dedicated, Fixed Function H/W
  - Performance & power-efficient, but weak flexibility
  - RPU [Woop, SIGGRAPH 2005]

- Fully Programmable Processor
  - Flexible, but inadequate performance and power consumption
  - Reconfigurable stream processor [Kim, CICC 2012] : 1~2 Mrays/sec
  - MIMD threaded processor [Spjut, SHAW-3 2012] : ~30 Mrays/sec
Requirements

• Performance for Real Time Rendering
  – 200~300Mray/sec

• Reasonable Flexibility
  – Programmable shading and ray generation
  – Support various BVHs: SAH/Binned/ SBVH/LBVH..
  – Easy to extend to GI (path tracing, photon mapping..)
  – Easy to combine rasterizer (OpenGL|ES) and ray tracing

• Low Power & Cost
SGRT
Our Approach

• Combination of CPU, H/W and DSP (Mobile SoC)
  – Tree Build: *sorting, irregular work* → Multi-core CPU (with multi-level $)
  – Refit, Traversal, Intersection: *embarrassingly parallel* → Dedicated H/W
  – Ray Gen. & Shading: *need for flexibility* → Programmable DSP
System Architecture

• SGRT (Samsung reconfigurable GPU based on Ray Tracing)
  – T&I Engine: fast, compact H/W to accelerate traversal & intersection
  – SRP: Samsung Reconfigurable Processor to support flexible shading
  – SMK: Parallelization framework
T&I Engine : A MIMD H/W Accelerator

- Newly designed H/W Accelerator based on our previous work – *KDtree H/W engine* [Nah, SIGGRAPH ASIA 2011]
  - Single-ray-based MIMD architecture: Efficient processing for incoherent rays
  - Ray Accumulation Unit (RAU): Hardware multithreading
- **Optimized restart & short stack** algorithm
  - Adaptive restart trail [Lee, HPG 2012]
- **Early Intersection Test**
  - Reducing expensive ray-primitive IST test
Early (Two-Pass) Intersection Test

![Diagram of Early (Two-Pass) Intersection Test]

Conventional IST
- Ray-nodeAABB Test
- Ray-Primitive Test
- Traversal Unit
- Intersection Unit

Early IST
- Ray-nodeAABB Test
- Ray-primAABB Test
- Traversal Unit
- Intersection Unit

Talk, ACM SIGGRAPH 2012

SIGGRAPH 2012
Ray Accumulation Unit

- Specialized H/W multi-threading for latency hiding [Nah, 2011]
  - Missed rays are accumulated in RA buffer, other rays can be processed during this period
  - Coherence can be increased, the rays that reference the same cache line are accumulated in the same row in an RA buffer
  - Experimental results, up to 3x performance gain
Samsung Reconfigurable Processor

- A flexible architecture template [Lee, HPG 2011/2012]
- ISA such as arithmetic, special function and texture are properly implemented.
- The VLIW engine useful for GP computations (function invocation, control flow).
- The CGRA makes full use of software pipeline technique for loop acceleration.
Packet Stream Tracing on SRP

- Remove recursion
  → Job-Q based streamed iteration

- Classified according to the types of operation → CGA kernel

- A packet of rays are batched

- Each kernels are mapped on CGA, loop accelerated
  - shows high IPC rate up to the maximum number of FU arrays
Parallelization Framework

- Parallel ray tracing with multi-tasking system
  - Utilized embedded RTOS, SMK (Samsung Multi-Platform Kernel) [Shin, SAC 2011]
  - Supports multi-tasking by systematic scheduling in the task queues

- Individual task for each SGRT core is responsible for
  - Different pixels (or pixel tiles), the scheduler can distribute the next tasks to the idle SGRT core first, $\rightarrow$ dynamic load balancing
Evaluation
Simulation Environment

- Built a cycle accurate simulator (T&I Engine), and a in-house cycle accurate compiled simulator, called csim (SRP)
- Test condition w/ two benchmarks
  - Full SAH, cost ratio 5:1 (TRV:IST) for shallow tree
  - Ferrari scene (210K triangles, 1 light source)
  - Fairy scene (170K triangles, 2 light sources)
  - Shadow, reflection, refraction @WVGA (800x640)
Preliminary Results

• Architecture configuration
  – 4 SGRT cores, traversal & intersection unit = 4:1 per SGRT core
  – 1Ghz core clock

• Achieved around 170 MRPS (T&I), 255 MRPS (RGS) for Fairy
  – Recent GPU ray tracer (156~317 MRPS, NVIDIA Kepler) [Alia, HPG 2012]

<table>
<thead>
<tr>
<th>Scene</th>
<th># of tri.</th>
<th># of ray</th>
<th>T&amp;I Engine</th>
<th>SRP</th>
<th>Simulated FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pipeline usage</td>
<td>TRV $ hit ratio</td>
<td>IST $ hit ratio</td>
</tr>
<tr>
<td>Fairy</td>
<td>170K</td>
<td>1.7M</td>
<td>87.27</td>
<td>93.83</td>
<td>96.53</td>
</tr>
<tr>
<td>Ferrari</td>
<td>210K</td>
<td>1.5M</td>
<td>79.75</td>
<td>92.56</td>
<td>92.92</td>
</tr>
</tbody>
</table>
FPGA

- Currently, we are also testing the SGRT on FPGA board
Conclusion
Conclusion

• SGRT: A novel mobile GPU based on ray tracing,
  – first mobile GPU to realize a real-time ray tracing
• Carefully designed to suit for mobile SoC environment
• Currently implementing the T&I engine at the RTL level
• Future work
  – Analyze cost and power consumption
  – Support dynamic scenes with a fast BVH build algorithm
    optimized for mobile environment
  – Higher-level shading/ecosystem

• Poster (#103) session: 8/7, 8/8 12:15-13:15PM
Acknowledgements

• This project is based on the collaboration with two University (Yonsei, National Kongju). Authors appreciate to two professors (Tack-Don Han, Hyun-Sang Park) for their valuable advices.

• Thanks